Method and Apparatus for Programming Computing Devices

The present invention relates to a method and apparatus for programming computing devices, and also to computing devices when programmed.

All computing devices, including PC's, Apples, portables, small devices such as mobile telephones, EFT terminals, etc., require programming in order that they can operate to provide a function. A program is essentially a series of instructions which instructs the computing device to operate in a certain way. The programming may also include data or require data to be input so that the instructions can be carried out.

All modern programming requires an "initialisation" process. During initialisation in object programming for example, all the "objects" and variables are set up so that the program can then operate the device to provide the desired function. Programs are usually provided with code which enables initialisation, and may also require the existence of other programming on the device to assist with initialisation. Initialisation code is run in an identical manner every time the program is set up for use. The initialisation code takes up a finite amount of the computer memory and requires computer processing time. Once initialisation has been carried out, the program may operate normally. Any code which is not necessary for operation of the program but which was necessary for the initialisation is now redundant. It will be required again, however, when the program needs to be initialised again after the device has been switched off and when it is switched on again. It is common practice in the computer industry to initialise every time a device is switched on or a program used for the first time once the device has been switched on.

Note that the present invention is not limited to object-orientated programming. All programmed computing devices include programs which require an initialisation

process before the program can operate to perform its function. The present invention therefore has general application.

"Small" computing devices, such as electronic funds transfer (EFT) terminals, mobile phones personal organisers, etc, do not have a great deal of memory space to store program instructions. The functionality of these devices is therefore somewhat limited. The need to provide instructions for initialisation of programs further limits the memory space which is available for functionality of such devices.

The present invention provides a method of programming a computing device, comprising the steps of providing a complete program for operating the device to perform a function, initialising the complete program to produce a functional program which is in a form ready to operate the device to perform the function, and loading the device with the functional program.

The program loaded in the computing device is therefore already in an initialised form. Preferably, any part of the complete program which is no longer necessary for operation after initialisation (e.g., any code only necessary for initialisation) can be dispensed with. Also any other code which may not have been part of the complete program, but which may have been necessary for initialisation, is preferably not loaded into the computing device (e.g., in object programming, some "methods" are necessary to initialise a program - many of the methods are common to different programs however but are not needed afterwards).

Preferably, a separate computing device is used to perform the initialisation, and then the functional program is loaded into the computing device, from the separate computing device. This has the advantage that a larger, more powerful computing device can be used to initialise the program. The computing power of this large device can

therefore be used in the initialisation step (it does not rely on computing power of the small computing device) and the separate computing device may store various initialisation means (may store and run "methods" used to set up programs) separately from the complete program. This provides the option of being able to use big computer routines, requiring large computer power, in the initialisation phases and producing a functional program which is substantially smaller than the complete program, for loading into a device having small memory. Computer programs could be designed with this process in mind.

Advantageously, therefore, the present invention effectively enables a large program to be run in a small device, which may be an already existing device such as an EFT terminal, so that the small device can run more complex programs and have more functionality than if using present methods of programming which require the small device to contain sufficient program to perform initialisation each time the device is set up or switched on.

The present invention further provides a programming apparatus for programming a computing device, the programming apparatus including initialisation means for initialising a complete program to produce a functional program for operating the computing device to perform a function, the functional program being in a form ready to operate the device, and means for loading the computing device with the functional program.

The programming apparatus may be a general purpose computer such as a PC, Apple, Main Frame, etc. The computing device may also be any device but is preferably a smaller computing device, such as an EFT, mobile telephone, remote control.

The present invention yet further provides a computing device including a functional program arranged to operate the computing device to perform a function, the functional program being of a post-initialised form, having been

initialised before loading into the computing device.

The computing device may be any computing device and may be any general purpose computer. Preferably, however the computing device is a small device, by which we mean devices that are not usually considered to be general purpose computers (general purpose computers being PC's, main frames, Apples and similar devices) but are devices such as EFT terminals, mobile telephones, remote controls for other devices and other (usually dedicated) small devices.

They may also include small computers, such as palm-top type computers.

Preferably, the device includes only code required for the post-initialised functional program and any code required for initialisation is not included in the device.

Features and advantage of the present invention will become apparent from the following description of an embodiment thereof, by way of example only, with reference to the accompanying drawings, in which;

Figure 1 is a schematic block diagram of a generalised programming apparatus for programming a computing device, and a computing device, in accordance with an embodiment of the present invention;

Figure 2 is a schematic block diagram of an embodiment of a remote payment terminal which may be programmed in accordance with an embodiment of the present invention;

Figure 3 is a schematic diagram of a control program architecture for the payment terminal of figure 2, and

Figure 4 is a schematic diagram showing a structural embodiment of message instructions and description for the message processing means 105 of figure 3.

Referring to figure 1, a programming apparatus 200 in accordance with an embodiment of the invention is illustrated. The programming apparatus 200 may comprise a generalised computer, such as a PC, Apple, etc. The programming apparatus will include input means, such as a

keyboard and mouse, display means, such as a VDU and processing means (not shown). This apparatus is represented in the figure 1 diagram by "hardware" block 201. In accordance with the present invention, the programming apparatus will also include software represented by blocks 202 and 203. Of course, other software may be included in the apparatus 200, as it is a general purpose computer, but the diagram of figure 1 only shows software which is pertinent to the present invention.

Block 204 represents a computing device, which is preferably a small computing device such as an electronic funds transfer (EFT) terminal (remote payment terminal). The small computing device also includes hardware which is necessary to perform its (usually dedicated) function 205 and also software 206 for controlling the hardware 205 to perform the function. In the case of device 204 being a remote payment terminal, the function will be to deal with EFT (see later).

In a conventional device, as discussed above, the software 206 included in the smaller computing device will include not only code which is necessary to control the hardware 205 to perform the required function of the device, but also code and routines which are necessary to initialise the software on switch on of the device so that software is then ready to perform its function (post-initialisation). In the embodiment illustrated in figure 1, however, the software 206 is in the form of a functional program. This functional program is present in the device in an initialised state and no initialisation is required on switch-on. The program is already initialised. Further, this post-initialised program does not include any "excess" code or routines which would have been necessary for initialisation if it had been a conventional program. Functional program 206 can therefore provide greater functionality because more program can be stored in the hardware 205 memory than would have been the case if it was

also necessary to store code and routines for an initialisation process.

Any program, however, must go through an initialisation phase at some point before it can perform its function. In the present invention, the apparatus 200 is arranged to produce the post-initialised functional program 206 and load it into the device 204.

It does this by the following process:

In the programming apparatus 200, a complete program 203 including initialisation code is loaded. This complete program, when initialised, provides the functional program 206. Initialisation routines (which may be in the form of methods) 202 are also included in the programming apparatus 200. After the complete program 203 has been loaded, it is initialised in the device 200, using the initialisation routines (if any) and the computing power (which will be greater than the computing power of the smaller device 204) of the device 200. Following initialisation, any "excess code" not necessary in the functional program 206 is removed and the functional program 206 is loaded in the smaller device 204.

The smaller device 204 can therefore have greater functionality than it otherwise would have done if it needed to store complete program with initialisation code and routines.

The programming language used may be any language, but the present invention is particularly suited to object programming, using such languages as C++ or Java.

Figure 2 is a block diagram of a remote payment terminal device which may be programmed using the embodiment of figure 1, i.e., the terminal in figure 2 may be the device 204 of figure 1.

Such devices operate to facilitate remote payment transactions, and a general overview of operation is as follows:

(1) Information is taken from an account holder's

(customer) card 5 via a card reader 4. Transaction information is input via the keyboard 3. The transaction information may include a money amount. The display 6 may prompt the user (merchant employee, customer) to input information (e.g., it may ask a merchant employee to input an amount) and may also display information as it is input. The keyboard 3 may also be used by the customer to input a code for the account, such as a PIN number.

(2) The CPU communicates the information via communications interface 8 with an account acquirer computer. The account acquirer computer may carry out a transaction (e.g., deduct money from the customers account and pay the merchants account) or may provide an "authorisation" that a transaction can be carried out. Information that an account transaction has taken place or that the account acquirer authorises a transaction to take place is transmitted to the communications interface 8 from the account acquirer computer. A display 6 may be provided to indicate that the transaction has occurred or may proceed.

(3) When the transaction is complete, a print out of transaction information may be provided from printer 7.

Prior art payment terminal devices are generally programmed in a conventional manner. That is, programming comprises a sequential set of operating instructions (including an initialisation sequence) which are executed in sequence to carry out a remote payment transaction. This "sequential program" may be directly compiled onto the processor of the device so that the device is under direct program control or, as is more usual, an applications program in a conventional programming language may control operations through a BIOS/OS. Applicants prior patent application International Application No. PCT/AU98/00173, the disclosure of which is incorporated herein by reference, discloses a method of programming small computing devices such as remote payment terminals, which

allows the programs to be portable between devices having different architectures, by utilising a "virtual machine". The virtual machine is a computer program to emulate a hypothetical computer. Different incompatible computers may be programmed to emulate the same hypothetical computer. Any computer programmed to emulate the hypothetical computer will thus be capable of executing programs for the virtual computer. This creates a complete portable environment for program operations. The problem with virtual machines, is that operation is generally slower than with a device which is programmed specifically for its architecture. The applicants earlier patent application overcomes this problem with EFT machines and other communications devices by creating a virtual machine which includes specialised software "processors" for carrying out communications and organisation of communications functions. A more detailed description is given in the application referenced above, but for the purposes of the present application, this type of programming is well suited to applying a programming method in accordance with an embodiment of the present invention.

Figure 3 is a schematic block diagram illustrating architecture of a device utilising the type of programming disclosed in PCT/AU98/00173.

The architecture comprises the hardware 100 of the device, as illustrated and described in relation to figure 1. It also comprises the hardware drivers, known in the prior art, and including an existing BIOS/OS or hardware drivers, reference numeral 101 and also includes the Hardware Abstraction Layer Interface (HAL) 102. The HAL 102 and hardware drivers 101 form a layer of a virtual machine which also includes virtual machine processors 103.

The virtual machine 101, 102, 103 is arranged to emulate a hypothetical payment terminal. Application 104 controls the virtual machine 101, 102, 103 which in turn controls operation of the hardware 100. The virtual

machine 101, 102, 103 can be adapted for many different hardware 100 arrangements (i.e. many different brands of payment terminal). Different arrangements of hardware 100 can therefore be controlled by the same application software 104.

The provision of Hardware Abstraction Layers and hardware drivers for virtual machines is known in the prior art and fully described in various publications. Each peripheral of the virtual machine is defined to be able to act in some manner on a standard set of commands. The HAL implements the best interpretation of each command on the actual peripheral present. For example a printer is defined to implement a "feed paper ready for tear off" instruction. On differing roll paper printers this requires feeding a different number of lines, on tractor feed printers this requires feed to the next perforation.

The virtual machine processors, according to PCT/AU98/00173, include a message processor 105 and a protocol processor 106, implemented in software code. The message processor is arranged to process messages communicated to or to be communicated from the payment terminal via the communications interface 8. The protocol processor is arranged to organise communications to and from the device, and to control and select the sequence of message processor operations in relation to messages received and transmitted. The message processor 105 and protocol processor 106 are implemented in native code of the payment terminal and therefore operate at relatively high speed. Because much of the "work" of the payment terminal is in building, comparing and deconstructing messages and processing communications, the operation of the device is relatively quick even though employing a virtual machine, 101, 102, 103.

The virtual machine processors 103 also comprise a function processor 107 the operation of which is to control and select general operations of the device not specially

controlled by the message and protocol processors 105, 106. The function processor is also preferably implemented in the native code of the micro-processor of the hardware 100.

The application 104 includes protocol instructions 108, message instructions, 109, function support 110 and function instructions 111. The protocol instructions 108 govern operation of the protocol processor 106. The message instructions 109 provide directions for operation of the message processor 105. Function support 110 and function instructions 111 govern operation of the function processor 107. The application 104 and virtual machine 101, 102, 103 operate on data 112 input to the payment terminal to process it in accordance with the application 104.

In a preferred embodiment of PCT/AU98/00173, the virtual machine processes 103 are constructed using C and the application is constructed using C++ or Java. A detailed description of operation of the virtual machine and processors is given in the above-referenced application and is not necessary for the description of the embodiment of the present invention.

Figure 4 is a schematic diagram illustrating the structure of the message instruction means 109 (Figure 3). The message instruction means is in fact in the form of a set of "descriptors" of the messages. Each message usually comprises a plurality of fields 120, and the message instruction means for each message contains a corresponding plurality of message instructions. One field may be the CUSTOMER NAME, for example. In the message instruction means, each field is associated with a number of message descriptors 121 which designate characteristics to be applied to the information in that field or to be expected of the information in that field. Operations which may be carried out on the data included in that field may also be included in the descriptors 121. As illustrated in the drawing, the descriptors may include:

1. Data Location Identification. This will indicate either where the data is to be found and/or where data is to be put. The data location information is contained in a two byte field descriptor (thus having 65535 different possible values) with value ranges allocated to

1) 2000 strings

2) literal numeric values from 0 to 32,000 in abbreviated form

3) data field Ids where each ID is represented as an entry in a table, and each table may contain up to 256 fields.

2. Data Representation (i.e. Ascci, Binary, etc.). This indicates what representation form the data is in and/or what it is to be converted to.

3. Format. This provides a description of the format that the data is in and/or is to be placed in.

4. Test Function. The index of a function processor set of instructions to determine if the current field is to be included or excluded at this time

5. Line & Column. Relative position for use in constructing messages for display or printing. These values are used to determine the quantity of space characters, and or new line characters that are required in the buffer.

6. Substitution list. A list of text representations to substitute for numeric values e.g., display the value "1" as "Monday" and "2" as "Wednesday".

7. Additional description options as required by the application or prove useful in future embodiments.

Each message instruction will therefore include a description of a field of message data, providing instruction for the virtual message processor means which enable it to carry out a number of tasks:

1. To compare a message with a message description to see if it is the correct required message.

2. To take a message of the correct description from

a location and place it in an-other location.

3. To take a message and deconstruct it into various components and place the various components into other locations.

4. To take data and build a message in accordance with the message description and place the built message in a location.

5. Compare one message with another message.

Other functions may also be carried out by the message processor as required by the application. The message processor can manipulate data in any desired way in accordance with descriptors provided by the message instructions. Messages comprising data can therefore be built, placed in locations, taken from locations, de-constructed with elements being placed in locations, etc. for subsequent operation on the data by the application. Any device which deals with significant amounts of messages in such form can therefore benefit from this arrangement.

Each message description is labelled so that it can be identified by the application, e.g. each message description may be numerically labelled.

To get the message description as 109 in the format that is required for the preferred embodiment of PCT/AU98/00173, a lot of "setting up" is required for the programming. This is conventionally done during an initialisation step. Utilising the present invention, this initialisation step is carried out in the programming apparatus 200. All unnecessary code and routines are then dispensed with and the functional program (with the message instructions already set up with the appropriate fields) is loaded into the remote payment terminal. This type of programming can therefore be used for EFT terminals, and is particularly applicable where existing hardware includes small amounts of memory which may not provide to allow all the functionality of the programming which would be desired. By reducing the amount of code that has to be

loaded, performance can be improved.

The present invention therefore lends itself particularly for use with existing hardware. There are many small computing device which already exist which could be reprogrammed with higher functionality using the present invention. Examples are remote payment terminals, remote controls, mobile telephones, and others. Any type of device can be programmed in accordance with the present invention, but communications devices are particularly suited.

The following program code is an example of initialisation code which can be dispensed with, in accordance with the present invention, before loading the actual operational code into a device. The following code creates a message having three fields. The following code is extraneous to actual operation.

MsgBuff.txt

```
Buffer MessageBuffer = new Buffer ();
Field1 = new Field();
Field1.type(ASCII);
Field1.size(30);
Field1.SkipCharacters(3);
MessageBuffer.append(Field1);

Field2 = new Field();
Field2.type(HEX);
Field2.size(4);
MessageBuffer.append(Field2);

Field3 = new Field();
Field3.type(NUM);
Field3.size(8);
Field3.format($$$$$$9.99)
MessageBuffer.append(Field3);
```

In application number PCT/AU98/00173, the applicant discusses the provision of a Specialised Network Access

Computer (SNAC).

With the advent of the Internet and other extensive communications networks, it is believed that the operation of computers, such as PC's, will become more and more oriented towards acting as "servers" and/or "browsers". In other words, a major function of PC's connected to a network will be to operate either as a server, providing information and/or programs to the network for access by other parties, or as a "browser" for obtaining information/programs available on the network and operating on them. It is likely, in fact, that PC's will be asked to operate as both a server and a browser. This operation will not merely be restricted to the Internet, but for any network, even Local Area Net-works.

The applicant also believes that many other classes of devices may be connected to a network. For example in the future a home video cassette recording machine could be connected to the Internet (along with other devices) allowing remote programming from a browser device. An example of the use of this would be a worker upon learning of a requirement to stay at the office late and miss a favourite show could access their home VCR from the office and pro-gram it.

Telephone calls will eventually be digital and most likely use the Internet as the digital network. Like the VCR, this does not mean all phones would need a qwerty keyboard and colour display. They will both represent other classes of Internet connected devices- not requiring the exact same configuration as PC's.

The present invention facilitates the production of a small, economical device which is particularly arranged to deal with communications, to build, compare and deconstruct message information. Such a device is novel and may be termed a Specialised Network Access Computer (SNAC). The applicants believe that a SNAC could emerge as a class of device allowing data entry and control through the Internet

where a smaller, more economical device than a conventional PC is appropriate. In a preferred embodiment, the device is implemented utilising a virtual machine having a message processor and a protocol processor as discussed above. In the preferred embodiment, the software of the device can be considered to include three layers of virtual machine software (the HW drive layer, the Hardware Abstraction Layer, and the Virtual Machine Processor layer) and a software application. A payment terminal can be used substantially without alteration as the hardware component of the device.

Such a SNAC can be applied in many different types of communication application over a network.

Programming in accordance with the present invention can be applied in order to increase the functionality of small devices.

It will be appreciated by persons skilled in the art that numerous variations and/or modifications may be made to the invention as shown in the specific embodiments without departing from the spirit or scope of the invention as broadly described. The present embodiments are, therefore, to be considered in all respects as illustrative and not restrictive.